Period recovery under a survey window function

The alias family is a multimodal posterior, not a wrong answer.

This notebook walks through the analysis end to end on a *real* ATLAS cadence, showing the time-domain representation at each step: the raw epoch series, the spectral window, the periodogram, and the phase fold at every candidate period.

The forward operator is `A = S∘F_P` — fold at trial period `P`, then sample at the measured epochs — with measured heteroscedastic per-epoch errors. It is linear in the folded light curve and nonlinear but differentiable in the period.

<style> div.jp-CodeMirrorEditor pre, div.highlight pre, pre, .jp-RenderedText pre, .jp-OutputArea-output pre { white-space: pre-wrap !important; overflow-wrap: anywhere; word-break: break-word; } .jp-Cell { max-width: 100% !important; } </style>

In [1]:

```
import sys, json, warnings
sys.path.insert(0, "..")
warnings.filterwarnings("ignore")

import numpy as np
import torch

# style.py sets the Agg backend on import (it is used by headless figure
# scripts), so switch to the inline backend AFTERWARDS or no figure is captured.
from timedomain.eval.style import COLORS, MUTED, use_style
use_style()
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.dpi"] = 110

DEV = "cuda" if torch.cuda.is_available() else "cpu"
print("device:", DEV)
```

```
device: cuda
```

1. The measured observing window

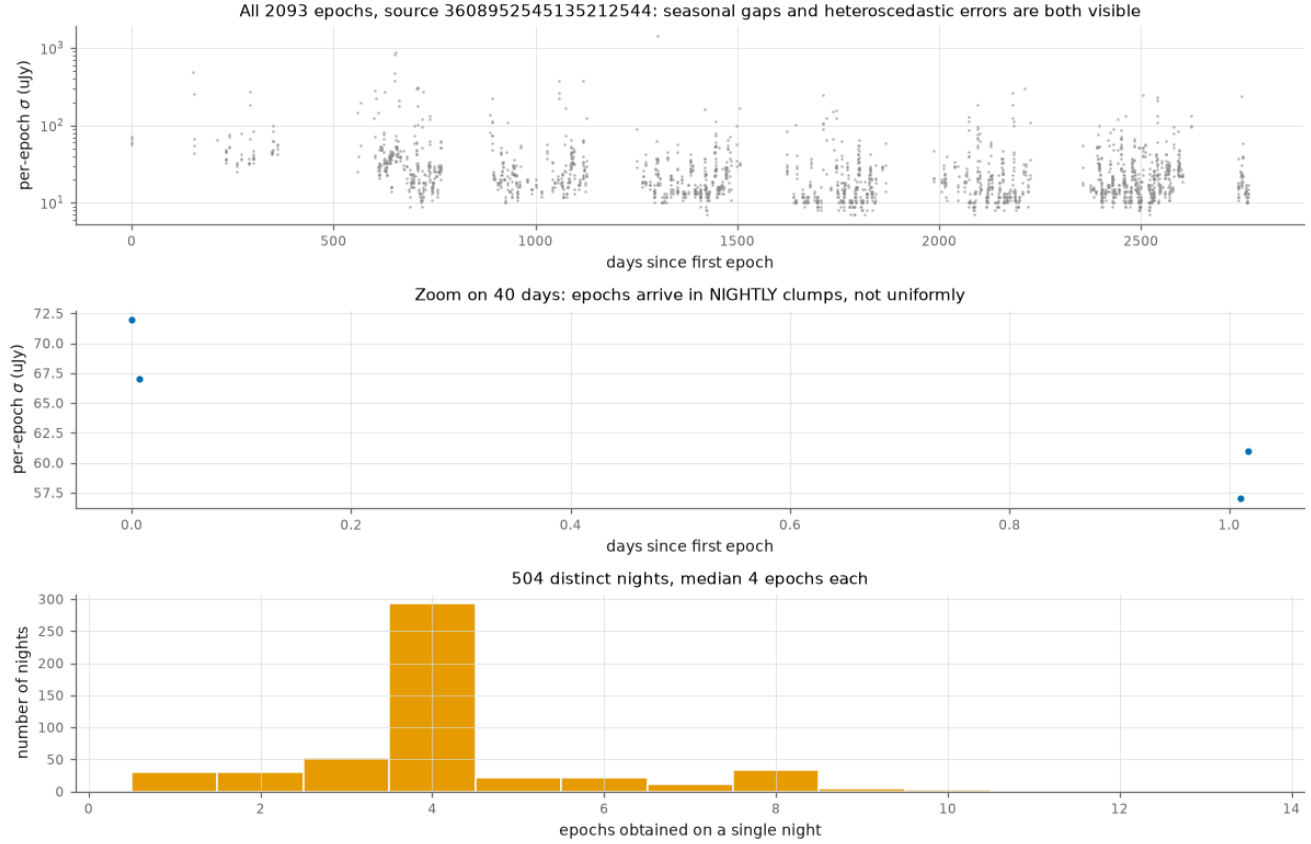
Everything starts from real data. These are ATLAS forced-photometry epochs and their per-epoch uncertainties — not an idealised cadence.

In [2]:

```
from timedomain.problem.cadence import load_bank, F_SIDEREAL, F_SOLAR

bank = load_bank()
print(f"{len(bank)} real ATLAS cadences")
for cad in bank[:5]:
    print(f"  {cad.source_id}: N={cad.n_obs:5d}  T={cad.baseline:7.1f} d  "
          f"1/T={cad.freq_resolution:.2e} c/d  median sigma=
{np.median(cad.sigma):.1f} uJy")

cad = bank[0]
```

```
t0 = cad.times.min()
fig, ax = plt.subplots(3, 1, figsize=(11, 7.2))

ax[0].plot(cad.times - t0, cad.sigma, ".", ms=1.5, color=COLORS["data"], alpha=.5)
ax[0].set_ylabel("per-epoch $\\sigma$ (uJy)")
ax[0].set_xlabel("days since first epoch")
ax[0].set_title(f"All {cad.n_obs} epochs, source {cad.source_id}: seasonal gaps "
                "and heteroscedastic errors are both visible", fontsize=10)
ax[0].set_yscale("log")

m = (cad.times - t0) < 40
ax[1].plot(cad.times[m] - t0, cad.sigma[m], "o", ms=3, color=COLORS["PnP-DM"])
ax[1].set_xlabel("days since first epoch")
ax[1].set_ylabel("per-epoch $\\sigma$ (uJy)")
ax[1].set_title("Zoom on 40 days: epochs arrive in NIGHTLY clumps, "
                "not uniformly", fontsize=10)

night = np.floor(cad.times - t0)
per_night = np.bincount(night.astype(int))
per_night = per_night[per_night > 0]
ax[2].hist(per_night, bins=np.arange(0.5, per_night.max() + 1.5),
           color=COLORS["DAPS"], edgecolor="white")
ax[2].set_xlabel("epochs obtained on a single night")
ax[2].set_ylabel("number of nights")
ax[2].set_title(f"{len(per_night)} distinct nights, median "
                f"{np.median(per_night):.0f} epochs each", fontsize=10)
plt.tight_layout(); plt.show()
```

```
24 real ATLAS cadences
  3608952545135212544: N= 2093  T= 2766.1 d  1/T=3.62e-04 c/d  median sigma=19.0 uJy
  2347123314285733376: N= 1900  T= 2741.5 d  1/T=3.65e-04 c/d  median sigma=19.0 uJy
  1159384163372832512: N= 1819  T= 2738.4 d  1/T=3.65e-04 c/d  median sigma=17.0 uJy
  3408191626098688800: N= 1765  T= 2734.8 d  1/T=3.66e-04 c/d  median sigma=18.0 uJy
  3464485547739672 32: N= 1926  T= 2734.8 d  1/T=3.66e-04 c/d  median sigma=16.0 uJy
```



2. The spectral window: the "dirty beam" of a time-domain survey

Sparse sampling in time convolves the truth with the spectral window $|W(f)|^2 = |\sum_k e^{-2\pi i f t_k}|^2$. Its sidelobes **are** the alias family. For ATLAS the peak positions obey a lattice measured independently:

$$f = a\, f_{\rm sidereal} + \frac{b}{4} f_{\rm solar}, \qquad a, b \in \mathbb{Z}$$

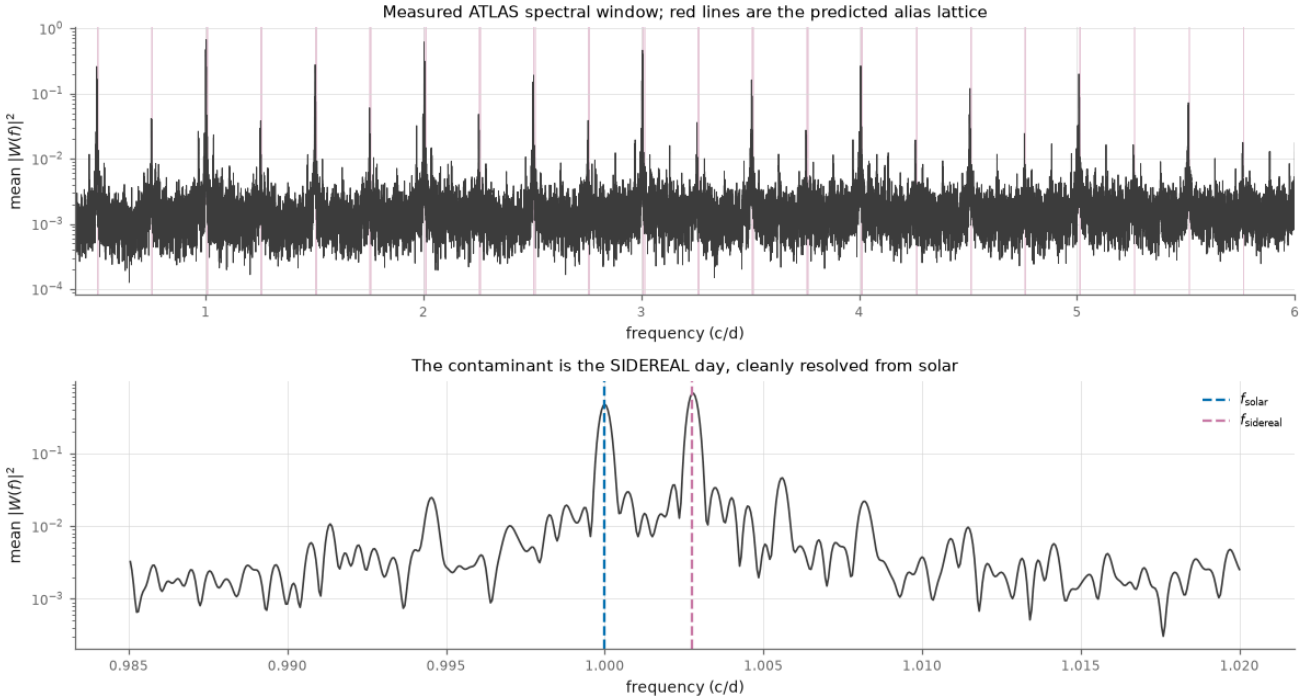
The quarter-integer solar term is the ~6 h spacing of the four-site ATLAS network.

In [3]:

```
res = json.load(open("../results/window_validation.json"))
print(f"{res['n_peaks']} peaks with contrast > 20 in 0.4-6 c/d")
print(f"median |residual| to the lattice: {res['median_abs_residual_cpd']:.2e}
c/d")
print(f"window width 1/T = {res['window_width_cpd']:.2e} c/d  ->  residual is "
      f"{res['median_abs_residual_cpd']/res['window_width_cpd']:.2f} x the width")
print(f"{res['frac_peaks_on_lattice']*100:.0f}% of peaks within 1/T of a lattice
line\n")
print(f"{'f (c/d)':>10s} {'P (min)':>9s}  lattice            {'contrast':>8s}")
for pk in res["peaks"][:8]:
    print(f"{pk['freq']:10.5f} {pk['period_min']:9.2f}  "
          f"{pk['a']} f_sid {pk['b']:+d}/4 f_sol   {pk['contrast']:8.0f}")

freqs = np.arange(0.4, 6.0, 5e-5)
win = np.zeros_like(freqs)
for cd in bank[:8]:
    win += cd.spectral_window(freqs)
win /= 8

fig, ax = plt.subplots(2, 1, figsize=(11, 6))
ax[0].plot(freqs, win, lw=.6, color="0.25")
for a_ in range(6):
    for b_ in range(-8, 9):
        f = a_ * F_SIDEREAL + (b_ / 4) * F_SOLAR
        if 0.4 < f < 6.0:
            ax[0].axvline(f, color=COLORS["LS"], lw=.5, alpha=.30, zorder=0)
ax[0].set_yscale("log"); ax[0].set_xlim(0.4, 6)
ax[0].set_xlabel("frequency (c/d)"); ax[0].set_ylabel(r"mean $|W(f)|^2$")
ax[0].set_title("Measured ATLAS spectral window; red lines are the predicted "
                "alias lattice", fontsize=10)

z = (freqs > 0.985) & (freqs < 1.02)
ax[1].plot(freqs[z], win[z], lw=1.1, color="0.25")
ax[1].axvline(F_SOLAR, color=COLORS["PnP-DM"], ls="--", label="$f_{\\rm solar}$")
ax[1].axvline(F_SIDEREAL, color=COLORS["LS"], ls="--", label="$f_{\\rm sidereal}$")
ax[1].set_yscale("log"); ax[1].legend(fontsize=8)
ax[1].set_xlabel("frequency (c/d)"); ax[1].set_ylabel(r"mean $|W(f)|^2$")
ax[1].set_title("The contaminant is the SIDEREAL day, cleanly resolved from solar",
                fontsize=10)
plt.tight_layout(); plt.show()
```

```
26 peaks with contrast > 20 in 0.4-6 c/d
median |residual| to the lattice: 6.40e-05 c/d
window width 1/T = 3.66e-04 c/d  ->  residual is 0.17 x the width
92% of peaks within 1/T of a lattice line
```

f (c/d)	P (min)	lattice	contrast
1.00278	1436.01	1 f_sid +0/4 f_sol	337
2.00276	719.01	1 f_sid +4/4 f_sol	300
3.00552	479.12	2 f_sid +4/4 f_sol	221
1.00002	1439.97	0 f_sid +4/4 f_sol	235
2.00552	718.02	2 f_sid +0/4 f_sol	149
4.00828	359.26	3 f_sid +4/4 f_sol	124
4.00554	359.50	2 f_sid +8/4 f_sol	123
1.50280	958.21	1 f_sid +2/4 f_sol	116



3. Inject a known signal on the real cadence

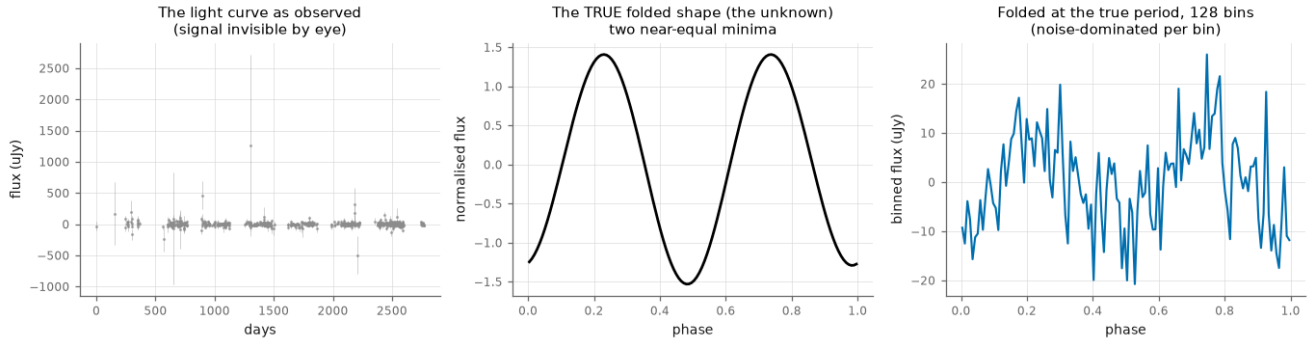
Synthetic signal, **real sampling and real errors**. We use a contact binary — two near-equal minima per period — because its `P/2` alias is *physically* ambiguous: folding at half the period produces a plausible single-humped curve. That makes it the honest test of whether a method reports ambiguity when ambiguity is real.

In [4]:

```
from timedomain.problem.dataset import load_benchmark

cases = load_benchmark()
case = next(c for c in cases if c.cls == "contact" and c.regime == "alias_pair")
print(f"case: {case.case_id}")
print(f"  true period {case.period_true:.5f} d, N={case.n_obs} epochs, "
      f"per-epoch SNR {case.snr}, total SNR {case.meta['total_snr']:.1f}")

ph_grid = (np.arange(128) + 0.5) / 128
fig, ax = plt.subplots(1, 3, figsize=(12.5, 3.4))
tt = case.times - case.times.min()
ax[0].errorbar(tt, case.flux, yerr=case.sigma, fmt=".", ms=2, lw=.4,
               color=COLORS["data"], alpha=.6)
```

```
ax[0].set_xlabel("days"); ax[0].set_ylabel("flux (uJy)")
ax[0].set_title("The light curve as observed\n(signal invisible by eye)",
fontsize=9.5)

ax[1].plot(ph_grid, case.signal_true, color=COLORS["truth"], lw=1.8)
ax[1].set_xlabel("phase"); ax[1].set_ylabel("normalised flux")
ax[1].set_title("The TRUE folded shape (the unknown)\ntwo near-equal minima",
                fontsize=9.5)

op = case.operator(n_phase=128)
fold32, filled = op.fold_bin(torch.as_tensor(case.flux)[None, :], case.period_true)
f32 = fold32[0].numpy()
ax[2].plot(ph_grid, np.where(filled[0].numpy(), f32, np.nan),
           color=COLORS["PnP-DM"], lw=1.4)
ax[2].set_xlabel("phase"); ax[2].set_ylabel("binned flux (uJy)")
ax[2].set_title("Folded at the true period, 128 bins\n(noise-dominated per bin)",
                fontsize=9.5)
plt.tight_layout(); plt.show()
```

```
case: contact_P0.28370_snr0.35_n600_3608952545135212544_alias_pair
  true period 0.28370 d, N=600 epochs, per-epoch SNR 0.35, total SNR 8.6
```



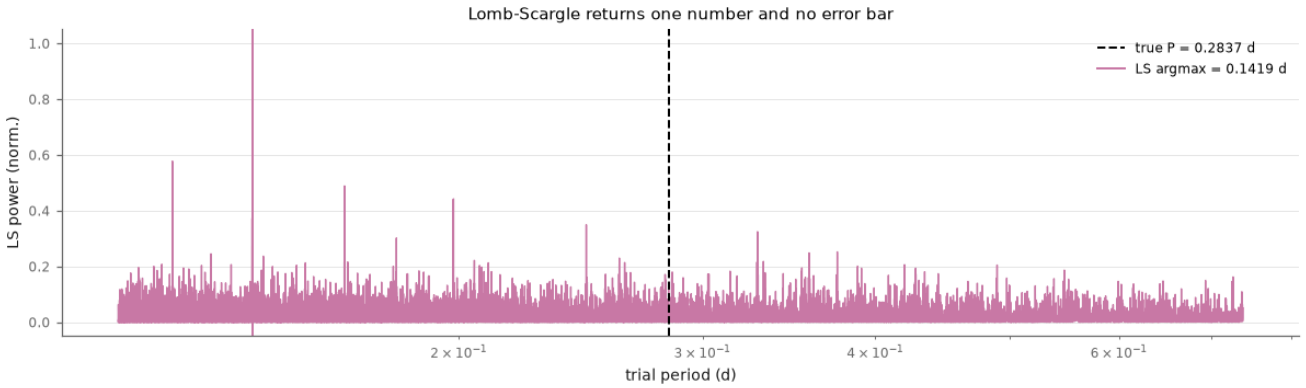
4. The classical baseline: Lomb–Scargle

A periodogram assumes a sinusoid, so it has no way to use the *shape* information that distinguishes a fold at `P` from a fold at `P/2`.

In [5]:

```
sb = json.load(open("../results/stage_b.json"))
rec = next(r for r in sb if r["case_id"] == case.case_id)
scan = np.load(f"../results/scan_{case.case_id}.npz")
P = case.period_true

fig, ax = plt.subplots(figsize=(11, 3.4))
ax.plot(scan["periods"], scan["ls_power"] / scan["ls_power"].max(),
        color=COLORS["LS"], lw=.9)
ax.axvline(P, color=COLORS["truth"], ls="--", lw=1.3, label=f"true P = {P:.4f} d")
ax.axvline(rec["ls_argmax"], color=COLORS["LS"], lw=1.3,
           label=f"LS argmax = {rec['ls_argmax']:.4f} d")
ax.set_xscale("log"); ax.set_xlabel("trial period (d)")
ax.set_ylabel("LS power (norm.)"); ax.legend(fontsize=8)
ax.set_title("Lomb-Scargle returns one number and no error bar", fontsize=10)
plt.tight_layout(); plt.show()

print(f"LS argmax            {rec['ls_argmax']:.5f} d   "
      f"({'correct' if rec['ls_correct'] else 'WRONG'}, "
      f"ratio to true = {rec['ls_argmax']/P:.3f})")
```

```
print(f"LS + ATLAS alias mask {rec['ls_masked_argmax']:.5f} d   "
      f"({'correct' if rec['ls_masked_correct'] else 'WRONG'})")
print(f"prior chi2 argmin    {rec['prior_argmin']:.5f} d   "
      f"({'correct' if rec['prior_correct'] else 'WRONG'})")
```




```
LS argmax            0.14185 d   (WRONG, ratio to true = 0.500)
LS + ATLAS alias mask 0.14185 d   (WRONG)
prior chi2 argmin    0.14185 d   (WRONG)
```

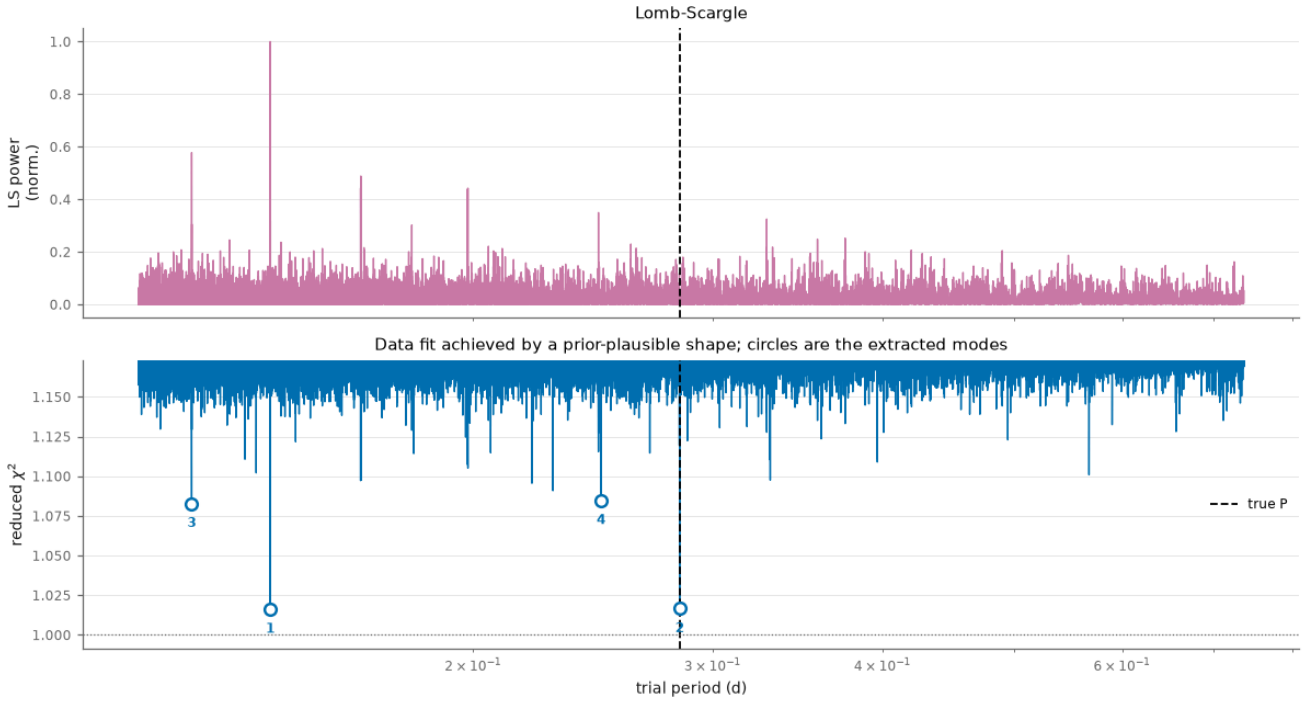
5. The data-fit landscape under a learned prior

For each trial period: fold, denoise the fold with a diffusion prior over light-curve shapes, and record the χ^2 that prior-plausible shape achieves. A fold at the wrong period is not a plausible light curve, so the prior cannot make it fit.

Two measures on different scales get **two stacked panels sharing the period axis** — never a twinned y-axis.

In [6]:

```
fig, ax = plt.subplots(2, 1, figsize=(11, 6), sharex=True)
ax[0].plot(scan["periods"], scan["ls_power"] / scan["ls_power"].max(),
           color=COLORS["LS"], lw=.9)
ax[0].set_ylabel("LS power\n(norm.)")
ax[0].axvline(P, color=COLORS["truth"], ls="--", lw=1.2)
ax[0].set_title("Lomb-Scargle", fontsize=10)

c2 = scan["chi2_prior"]
ax[1].plot(scan["periods"], c2, color=COLORS["PnP-DM"], lw=.9)
ax[1].axvline(P, color=COLORS["truth"], ls="--", lw=1.2, label="true P")
ax[1].axhline(1.0, color=MUTED, ls=":", lw=.8)
for i, m_ in enumerate(rec["modes"][:4]):
    ax[1].plot([m_["period"]], [m_["chi2_red"]], "o", ms=7, mfc="white",
               mec=COLORS["PnP-DM"], mew=1.6, zorder=5)
    ax[1].annotate(str(i + 1), (m_["period"], m_["chi2_red"]),
                   textcoords="offset points", xytext=(0, -14), ha="center",
                   fontsize=8, fontweight="bold", color=COLORS["PnP-DM"])
top = np.percentile(c2, 55)
ax[1].set_ylim(c2.min() - .16 * (top - c2.min()), top)
ax[1].set_xscale("log"); ax[1].set_xlabel("trial period (d)")
ax[1].set_ylabel(r"reduced $\chi^2$"); ax[1].legend(fontsize=8)
ax[1].set_title("Data fit achieved by a prior-plausible shape; circles are the "
                "extracted modes", fontsize=10)
plt.tight_layout(); plt.show()
```



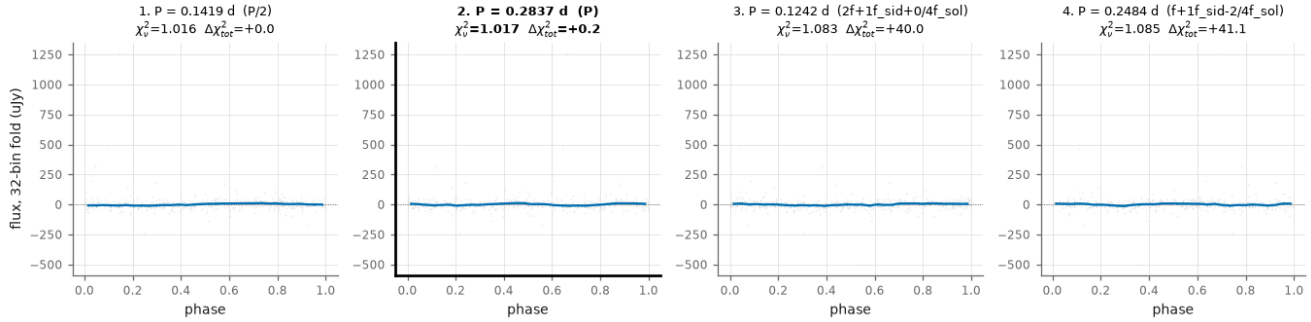
6. Fold the same data at each candidate mode

This is the crux. Each panel is the **same photometry**, folded at a different candidate period. Degeneracy must be judged on **total** χ^2: over $N \approx 600$ epochs a 10% difference in *reduced* χ^2 is a difference of 60 in total χ^2 — decisive, not a tie. Only gaps of a few units are genuine ties.

In [7]:

```
from timedomain.eval.metrics import mode_degeneracy
from timedomain.period.posterior import fold_bin_batch

deg = mode_degeneracy([m["chi2_red"] for m in rec["modes"]], rec["n_obs"])
modes = rec["modes"][:4]
M_DISP = 32          # display bins: at this SNR, 128 bins render as noise
phd = (np.arange(M_DISP) + 0.5) / M_DISP
tT = torch.as_tensor(case.times); yT = torch.as_tensor(case.flux)
sT = torch.as_tensor(case.sigma)

fig, axes = plt.subplots(1, len(modes), figsize=(3.1 * len(modes), 3.2))
for i, (axx, m_) in enumerate(zip(axes, modes)):
    pm = torch.tensor([m_["period"]], dtype=torch.float64)
    fold, den = fold_bin_batch(tT, yT, sT, pm, m=M_DISP)
    fv = fold[0].numpy(); cov = (den[0].numpy() > 0)
    err = np.where(cov, 1 / np.sqrt(np.maximum(den[0].numpy(), 1e-30)), np.nan)
    axx.scatter((case.times / m_["period"]) % 1.0, case.flux, s=1, alpha=.15,
                color=COLORS["data"], linewidths=0)
    axx.errorbar(phd, np.where(cov, fv, np.nan), yerr=err, lw=1.5,
                 color=COLORS["PnP-DM"], elinewidth=.7)
    axx.axhline(0, color=MUTED, ls=":", lw=.6)
    is_true = abs(m_["period"] / P - 1) < 0.02
    dchi = deg["delta_chi2_total"][deg["order"].index(rec["modes"].index(m_))]
    axx.set_title(f"{i+1}. P = {m_['period']:.4f} d  ({m_['label']})\n"
                  f"$\\chi^2_\\nu$={m_['chi2_red']:.3f}  "
                  f"$\\Delta\\chi^2_{{tot}}$={dchi:+.1f}",
                  fontsize=8.5,
                  fontweight="bold" if is_true else "normal",
```

```
                  color=COLORS["truth"] if is_true else "black")
    axx.set_xlabel("phase")
    if i == 0:
        axx.set_ylabel(f"flux, {M_DISP}-bin fold (uJy)")
    if is_true:
        for sp in axx.spines.values():
            sp.set_edgecolor(COLORS["truth"]); sp.set_linewidth(1.8)
plt.tight_layout(); plt.show()

print(f"{deg['n_degenerate']} modes lie within delta chi2_total <= 9 (3 sigma)")
print(f"top two differ by {deg['delta_chi2_top2']:.2f} in TOTAL chi2 over "
      f"{rec['n_obs']} epochs")
```




```
2 modes lie within delta chi2_total <= 9 (3 sigma)
top two differ by 0.24 in TOTAL chi2 over 600 epochs
```

7. Calibration: ambiguous where the physics is ambiguous

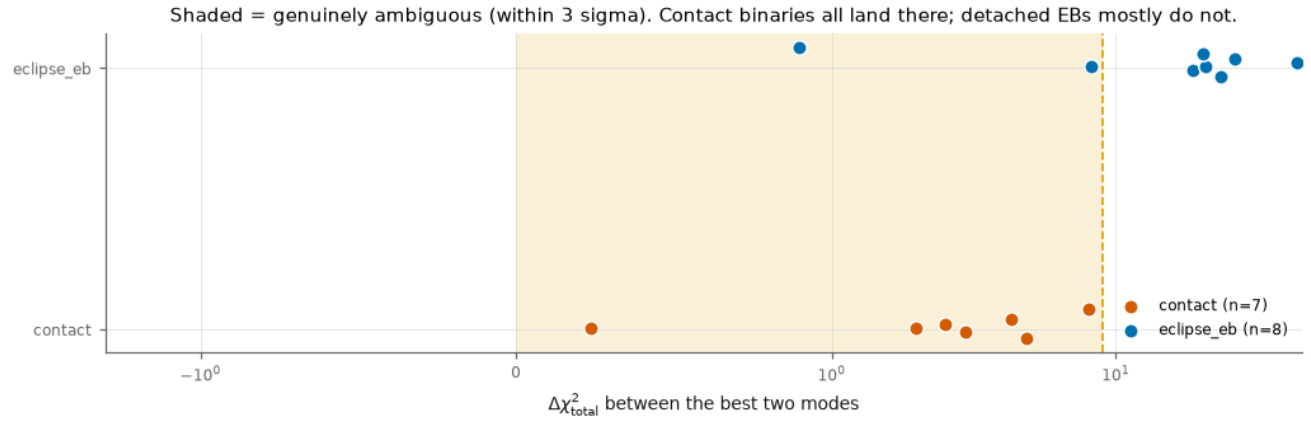
Morphology decides whether `P/2` is breakable *at all*. Unequal eclipse depths carry the information; near-equal minima do not. A calibrated posterior has to track that — and it does.

In [8]:

```
nb = [r for r in sb if r["regime"] != "ood"]
rows = []
for r in nb:
    d = mode_degeneracy([m["chi2_red"] for m in r["modes"]], r["n_obs"])
    rows.append((r["cls"], d["delta_chi2_top2"], d["is_degenerate"]))

print(f"{'class':<12s} {'cases':>6s} {'degenerate':>11s} {'median dchi2':>13s}")
for cls in ("contact", "eclipse_eb", "sine", "sawtooth", "flat_bottom"):
    rr = [x for x in rows if x[0] == cls]
    if rr:
        print(f"{cls:<12s} {len(rr):6d} {sum(x[2] for x in rr):>6d}/{len(rr):<4d} "
              f"{np.median([x[1] for x in rr]):13.1f}")

fig, ax = plt.subplots(figsize=(9.5, 3.2))
for cls, col in (("contact", COLORS["AdamL2"]), ("eclipse_eb", COLORS["PnP-DM"])):
    v = [x[1] for x in rows if x[0] == cls]
    ax.scatter(v, np.random.default_rng(0).normal(
        0 if cls == "contact" else 1, .06, len(v)), s=55, color=col,
        label=f"{cls} (n={len(v)})", edgecolors="white", zorder=3)
ax.axvspan(0, 9, color=COLORS["DAPS"], alpha=.14, lw=0)
ax.axvline(9, color=COLORS["DAPS"], ls="--", lw=1.1)
ax.set_xscale("symlog", linthresh=1)
ax.set_yticks([0, 1]); ax.set_yticklabels(["contact", "eclipse_eb"])
ax.set_xlabel(r"$\Delta\chi^2_{\rm total}$ between the best two modes")
```

```
ax.set_title("Shaded = genuinely ambiguous (within 3 sigma). Contact binaries all "
             "land there; detached EBs mostly do not.", fontsize=9.5)
ax.legend(fontsize=8, loc="lower right")
plt.tight_layout(); plt.show()
```

```
class          cases  degenerate  median dchi2
contact            7         7/7          3.0
eclipse_eb         8         2/8         20.4
sine               2         2/2          5.9
sawtooth           1         0/1         18.2
flat_bottom        1         0/1         19.8
```



8. Aggregate, and what it does and does not show

In [9]:

```
n = len(nb)
print(f"Exact period recovery over {n} in-distribution cases:")
print(f"  Lomb-Scargle argmax (astropy)      {sum(r['ls_correct_astropy'] for r in
nb)}/{n}")
print(f"  LS + measured ATLAS alias mask     {sum(r['ls_masked_correct'] for r in
nb)}/{n}")
print(f"  chi2 argmin under the prior        {sum(r['prior_correct'] for r in
nb)}/{n}")
print(f"  correct up to a HARMONIC (LS)      {sum(r['ls_harmonic_of_true'] for r in
nb)}/{n}")

sa = [r for r in json.load(open("../results/stage_a.json")) if "error" not in r]
print("\nSignal recovery at the true period (median over cases):")
print(f"  {'sampler':<10s} {'RMSE':>7s} {'chi2':>7s} {'two-sided chi2':>15s}")
for s in ("AdamL2", "DPS", "DAPS", "PnP-DM"):
    rr = [r for r in sa if r["sampler"] == s]
    if rr:
        print(f"  {s:<10s} {np.median([r['median_rmse'] for r in rr]):7.3f} "
              f"{np.median([r['median_chi2'] for r in rr]):7.3f} "
              f"{np.median([r['two_sided_chi2_at_median'] for r in rr]):15.3f}")
```

```
Exact period recovery over 19 in-distribution cases:
  Lomb-Scargle argmax (astropy)      5/19
  LS + measured ATLAS alias mask     4/19
  chi2 argmin under the prior        11/19
  correct up to a HARMONIC (LS)      19/19

Signal recovery at the true period (median over cases):
  sampler       RMSE    chi2  two-sided chi2
  AdamL2       0.911   0.786           1.272
  DPS          0.715   0.830           1.208
  DAPS         0.472   1.061           1.071
  PnP-DM       0.218   1.020           1.047
```

What this shows, and what it does not

Shows. The alias family is a genuinely multimodal posterior whose mode positions are predicted in advance by an independently measured instrument property. Contact binaries are 7/7 genuinely degenerate between `P` and `P/2` (median $\Delta\chi^2_{\mathrm{total}} = 3.0$) while detached eclipsing binaries are only 2/8 — the posterior is ambiguous exactly where the physics is ambiguous.

Does not show. The alias *mask* does not help (4/19 vs 5/19 unmasked): the dominant failure is the `P/2` harmonic, which is not a window alias and cannot be masked away. And the signals here are synthetic — the cadence and per-epoch errors are real, the light-curve shapes are not.

χ^2 is per measurement, and the unknown has 128 free bins against ~600 epochs, so values below 1 are absorbing noise rather than succeeding. That is why the two-sided statistic is reported alongside.